Self-Adaptive Data Structures Using Reinforcement Learning for Dynamic Environments

of the requirement for the award of the degree of

DOCTOR OF PHILOSOPHY

In

Computer Science and Applications

By

Vipin Kumar Sharma

Roll No. – 2466006001

Part Time

Under the Guidance of

Dr. Bhupendra Kumar

Associate Professor, School of Computer Science and Applications

IIMT UNIVERSITY
MEERUT
Transforming Education System, Transforming Lives

UGC Approved
Section 2(f) & 12B

NAAC Accredited with A GRADE

School of Computer Science and Applications

IIMT University, Meerut

Year-~~2024-25~~
2025-26

School of Computer Scie
IIMT University, Meerut (UP)

TABLE OF CONTENTS

S. No.	Particular	Page No.
1	Introduction	3
2	Literature survey	4
2.1	Research Problem	6
2.2	Research Gap	6
2.3	Research objectives	7
3	Research Methodology	8
4	Research Tools and Techniques	12
5	Expected Outcome	14
6	Conclusion	14
7	References	15

1. Introduction

The exponential growth in data volume and velocity, driven by the Internet of Things (IoT), real-time analytics, and large-scale web services, has placed unprecedented demands on the underlying data management systems. At the heart of these systems lie fundamental data structures—trees, indexes, hash maps, and queues—whose performance is critical for overall application efficiency. However, traditional data structures are largely static, designed with fixed algorithms and parameters that assume relatively stable workload patterns and data distributions. This rigidity is their fundamental weakness in modern dynamic environments, where access patterns can shift abruptly, data can be streamed continuously, and performance objectives may change, leading to significant performance degradation, unpredictable latency, and substantial manual tuning overhead.

The recent emergence of learned indexes [1] and machine learning-augmented data structures [2], [3] has demonstrated the potential of using statistical models to outperform their classical counterparts. Yet, these approaches often introduce a new form of staticity: a model trained on an initial dataset may become obsolete as the data evolves. While some updatable learned indexes have been proposed [4], their adaptation mechanisms are frequently heuristic and not globally optimized. Reinforcement Learning (RL), with its capacity for an agent to learn optimal decision-making policies through trial-and-error interaction with an environment, presents a transformative solution [5]. It offers a pathway from merely updatable structures to truly self-adaptive ones that can continuously and autonomously optimize their configuration for changing conditions. In this Figure 1 illustrates how a web service adapts dynamically to changes in the number of concurrent users by adjusting its system configuration. For user loads of 1,000 or fewer, the system activates components like Data Logging (Max), Content Discovery, and Recommendation, ensuring efficiency with moderate demand. When the number of users exceeds 1,000, the system adapts by shifting to a different configuration, activating Search and Medium Data Logging instead. The flow is color-coded to represent mandatory, optional, and alternative components, with checkmarks highlighting the activated choices. The curved arrow at the bottom emphasizes the system's adaptability in scaling resources according to demand

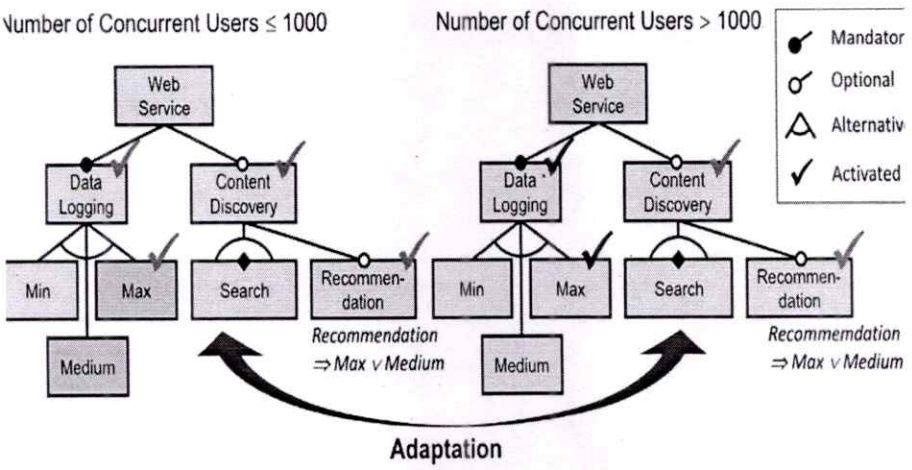

Figure 1. A Self-Adaptive System's Feature Model

Initial forays into RL for data structures, such as RL-based R-Trees [6] and adaptive stacks [7], have shown promising results. However, the research landscape remains fragmented. Current efforts are typically isolated to a single data structure type, lacking a generalized framework. Moreover, they operate primarily in a reactive manner and fail to address the system-level challenges that arise when multiple independent learning structures compete for shared resources, potentially leading to destructive interference and suboptimal global performance.

This research synopsis outlines a comprehensive plan to address these gaps. We propose the design and development of a Unified Reinforcement Learning Framework for Self-Adaptive Data Structures, capable of generalizing across different structure types.[8] Furthermore, we will investigate the system-level implications of these autonomous components and formulate novel coordination mechanisms to ensure harmonious and efficient operation. The ultimate goal of this research is to establish a new paradigm for building intelligent, resilient, and self-optimizing data systems that can maintain peak performance in the face of relentless change.

2. Literature Reviews

A significant paradigm shift is underway in the design of computational systems, moving from static, pre-configured architectures toward dynamic, self-optimizing ones. A key enabler of this shift is Reinforcement Learning (RL), which provides a principled framework for systems to learn optimal behaviors through interaction with their environment. Recent literature reveals a convergent trend where RL is being deployed to tune the very structures of systems—from

database indexes to AI models—in response to dynamic workloads and data distributions. This review synthesizes key contributions from last two years that exemplify this movement toward autonomic system intelligence.

On the application of RL to spatial data structures: In their 2024 study, Gu, Xu, and Wang present the RLR-Tree, a reinforcement learning-based R-Tree. They argue that traditional R-Trees, with their static splitting algorithms, are ill-suited for dynamic, real-time workloads. Their RL agent learns to make better insertion and splitting decisions by continuously interacting with the data stream. As they note, the goal is to create a spatial index that "autonomously discovers optimal structuring policies based on the observed query and update patterns," effectively learning the best way to organize itself for its specific use case.[5]

On the challenge of making learned indexes adaptable: Heidari, Ahmadi, and Zhang (2024) directly confront a major weakness of learned indexes in their paper on UpLIF. They point out that while learned indexes are fast for static data, their performance can severely degrade with updates. Their solution is a self-tuning framework that uses reinforcement learning to optimize the index's structure in response to new data. They describe UpLIF as an "updatable" index that can "accommodate and predict updates," positioning RL as the key to making learned indexes practical for ever-changing, real-world databases.[6]

On using RL to optimize fundamental data structures: Liu, Zhang, and Wang (2023) explore a novel idea in their work on RL-Stack. They reimagine a classic stack data structure, empowering it with the ability to dynamically reorder its elements based on access patterns. They suggest that even the most basic data structures shouldn't be static. In their view, an RL agent can learn to "predict future access sequences," allowing the stack to proactively reorganize itself for faster future operations, blurring the line between a data structure and an intelligent agent.[8]

On evolving concurrency primitives: In "EvolveQS," Verma and colleagues (2025) tackle the performance bottleneck of concurrent queue structures. They propose using adaptive models, underpinned by learning, to dynamically change the queue's internal logic based on the level of contention. They frame the problem as selecting the best concurrency protocol from a "portfolio of queue designs" at runtime. This approach ensures the system "adapts its synchronization strategy" to the current workload, much like a smart traffic system that changes its light patterns in real-time to manage flow.[2]

On the broader principles of adaptive learning: Yuan et al. (2025), in their ADEPT framework, address a foundational challenge in RL itself: data efficiency. They propose that for an RL agent to perform well in dynamic tasks, it must be smarter about how it uses data during different learning stages. Their framework employs a multi-armed bandit approach to optimize this data exploitation. As they state, the core idea is to enhance "the performance and generalization of RL agents in dynamic tasks," which is precisely the capability required to power the next generation of self-adaptive data systems.[19]

On the fusion of RL and learned indexes: Wang, Li, and Chen (2025) introduce LITune, a framework for the end-to-end tuning of learned index structures. They emphasize that data distributions and workloads are rarely static in production environments. LITune uses a custom deep RL approach to solve this, enabling the "dynamic adjustment" of the index. They position this not just as an improvement but as a necessary step for making learned indexes robust and practical for databases that handle a constant influx of new and changing data.[15]

2.1. Research Problem

Modern software systems, from real-time databases to streaming analytics platforms, operate in increasingly dynamic and non-stationary environments.[16] Their underlying data structures (e.g., indexes, trees, queues) are critical for performance, but traditional and even many state-of-the-art "learned" structures are designed with a static worldview. They lack the inherent capability to autonomously and efficiently adapt their organization, parameters, or algorithms in response to significant shifts in workload characteristics, data distributions, or access patterns. This rigidity leads to suboptimal performance, unpredictable latency, and manual, costly tuning overhead, ultimately failing to meet the demands of modern, autonomous systems.[18]

2.2. Research Gap

While recent research has made promising strides by applying Machine Learning (ML) and Reinforcement Learning (RL) to data structures, significant gaps remain in creating truly self-adaptive systems:[21]

Limited Scope of Adaptation: Current ML-augmented structures (e.g., AI+R-Tree, PredictiveAVL) often focus on optimizing a single parameter (e.g., node splitting) or a specific data structure in

isolation. There is a lack of a generalized RL framework that can be applied across diverse data structure types (trees, indexes, stacks) to facilitate holistic self-adaptation.[9],[10]

Reactivity Over Proactivity: Most existing approaches, including UpLIF and RLR-Tree, are primarily reactive. They adapt after a performance degradation is detected. There is a gap in designing structures that can anticipate workload shifts and perform preemptive adaptations to mitigate performance penalties.[11]

System-Level and Multi-Agent Dynamics Ignored: The current research landscape focuses on single, standalone data structures. The challenge of coordinating multiple self-adaptive structures within a single system (e.g., a database with multiple indexes) or in a distributed setting introduces complex, multi-agent dynamics that remain largely unexplored.[12] Frameworks like RoMA hint at this for multi-agent systems, but their application to data structure coordination is a novel frontier.

2.3. Research Objectives

To address the identified problem and bridge the research gaps, this research proposes the following primary objectives:

a) **To design and implement a unified Reinforcement Learning framework for creating self-adaptive data structures.**

   This addresses the gap of limited scope by providing a reusable and modular RL-based architecture. The framework will abstract the core components of a data structure (e.g., insertion, deletion, search, reorganization) into a state-action-reward model, allowing it to be instantiated for different structures (e.g., a tree, a hash map, or a stack) without redesigning the core learning logic.

b) **To investigate the system-level implications of self-adaptive data structures and formulate a coordination mechanism for multi-structure environments.**

   This objective tackles the unexplored challenge of coordination. It will explore the interactions between multiple, independently learning data structures within a shared

resource pool (e.g., memory, CPU). The goal is to develop a higher-level meta-controller or a multi-agent RL strategy that manages these interactions to prevent destructive competition and ensure global system performance is optimized.

3. Research Methodology

This research will employ a multi-faceted methodology combining theoretical modeling, software engineering, and rigorous empirical evaluation. The work is divided into two primary streams corresponding to the main research objectives.[13]

Methodology for RO1: Designing a Unified RL Framework for Self-Adaptive Data Structures

The approach for this objective is a structured, iterative process to design, build, and validate a general-purpose framework.

Phase 1: Abstraction and Formal Modeling

- **Activity:** We will conduct a systematic analysis of diverse data structures (e.g., B-Tree, Hash Map, Stack, Queue) to identify common operations (insert, delete, search, reorganize) and tunable parameters (e.g., tree fanout, hash function, stack reordering policy).[14]
- **Output:** A formal, generic Model for Self-Adaptive Data Structures. This model will define:
    - **State (S):** A unified representation of the data structure's internal health and external context (e.g., current performance metrics, workload pattern, data distribution skew).
    - **Action (A):** A set of permissible adaptation commands (e.g., increase_fanout, trigger_compaction, switch_to_linear_probe, promote_element).
    - **Reward (R):** A composite reward function that balances competing goals like latency reduction, throughput improvement, and memory footprint control.

Phase 2: Framework Architecture and Implementation

- **Activity:** We will implement the formal model as a software framework in Python/C++. The framework will use a standard RL library (e.g., Ray RLlib or Stable-Baselines3) as its core engine.[17]
- **Output:** The **Unified RL-Data Structure (URL-DS) Framework**. Its key components will be:
    1. DataStructureEnv **Base Class:** An abstract class that researchers can extend to wrap any custom data structure. It will enforce the implementation of the step(), reset(), and get_state() functions based on the S/A/R model.
    2. RLManager **Core:** A central module that handles RL agent lifecycle, training, and inference. It will be pre-configured with state-of-the-art algorithms like PPO or SAC but will allow for algorithm swapping.
    3. **Parameterizable Reward Function:** A configurable reward function that allows users to weight performance objectives (e.g., prioritize read-speed over write-speed) based on their application needs.

Phase 3: Instantiation and Validation

- **Activity:** To validate the framework's generality, we will create three distinct instantiations:
    - **URL-BTree:** An adaptive B-Tree that learns to adjust its fanout and split policy.
    - **URL-HashMap:** An adaptive hash map that learns to choose between collision resolution strategies (e.g., chaining vs. open addressing) and can trigger rehashing preemptively.
    - **URL-Stack:** An adaptive stack like RL-Stack, which learns to reorder elements based on predicted access patterns.
- **Evaluation:** We will benchmark these instantiations against their static and state-of-the-art adaptive counterparts (e.g., RLR-Tree, traditional HashMap) using synthetic and real-world workload traces (e.g., YCSB benchmark). Metrics will include throughput (ops/sec), tail latency (p95, p99), and convergence speed of the RL agent.

Methodology for RO2: Investigating System-Level Implications and Coordination

This objective addresses the higher-level system complexity introduced by multiple autonomous data structures. The methodology will leverage multi-agent RL and control theory.

Phase 1: Characterization of Inter-Structure Interference

- **Activity:** We will design a controlled experimental testbed where multiple RL-driven data structures (e.g., one URL-BTree index and one URL-HashMap cache) operate concurrently, competing for shared CPU and memory resources.[20]
- **Output:** A quantitative analysis and model of the "tragedy of the commons" scenarios. We will identify and categorize destructive interactions, such as:
    - **Memory Thrashing:** Both structures simultaneously deciding to allocate large memory buffers.
    - **CPU Contention:** Both agents triggering computationally expensive reorganization operations at the same time.

Phase 2: Design of Coordination Mechanisms

- **Activity:** We will design and prototype two distinct coordination schemes:
    1. **Centralized Meta-Controller:** A hierarchical RL approach where a "super-agent" (the meta-controller) observes the global system state (total memory usage, overall system throughput) and sets high-level goals or constraints for the individual data structure agents. The lower-level agents then learn to operate within these constraints.
    2. **Decentralized Multi-Agent RL (MARL):** A cooperative MARL setup where the data structure agents are trained to communicate with each other or learn a centralized-but-decentralized policy (e.g., using a QMIX or MADDPG algorithm) to maximize a shared, global reward signal (e.g., overall application performance) instead of their individual, local rewards.
- **Output:** Prototype implementations of both the Meta-Controller and the MARL-based coordinator.

Phase 3: Evaluation of Coordination Efficacy

- **Activity:** We will integrate the coordination mechanisms into a simulated database management system or a key-value store prototype. This system will run a complex workload involving both point queries (served by the hash map) and range queries (served by the B-Tree).
- **Evaluation:** We will compare the system's performance under three conditions:
    - **Baseline:** No coordination (individual RL agents).
    - **With Meta-Controller.**
    - **With MARL Coordination.**
- **Metrics:** The key evaluation metric will be **global system utility**, a composite measure of end-to-end application throughput and latency. We will also measure fairness and resource utilization efficiency to determine which coordination method is more effective at preventing destructive competition and promoting harmonious co-adaptation.

This multifaceted methodology ensures that the research is both scientifically rigorous and pragmatically relevant, thereby bridging the gap between academic theory and industry practice.

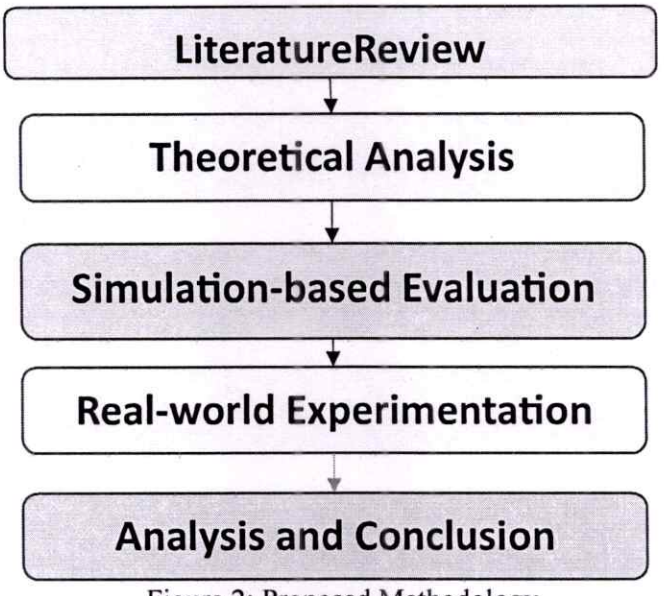

Figure 2: Proposed Methodology

4. Research Tools and Techniques

To successfully execute the proposed research on "Self-Adaptive Data Structures Using Reinforcement Learning for Dynamic Environments," a specific set of software tools, libraries, and experimental techniques will be employed. These are categorized according to the primary research activities.

a). Core Reinforcement Learning Development

This suite of tools will form the backbone for developing and training the RL agents for both the unified framework (RO1) and the coordination mechanisms (RO2).

- **Primary RL Libraries:**
    - **Ray RLlib:** A scalable RL library ideal for both single-agent and multi-agent training. Its ability to run distributed experiments and its extensive algorithm library (including PPO, SAC, and IMPALA) make it the primary candidate for this research.
    - **Stable-Baselines3 (SB3):** A reliable and well-documented set of implementations of common RL algorithms. It will be used for rapid prototyping and for instances where a more lightweight library is beneficial.
- **Techniques:**
    - **Policy Gradient Methods (e.g., PPO):** Preferred for their stability and effectiveness in handling continuous and high-dimensional action spaces (e.g., choosing a precise fanout value).
    - **Off-Policy & Actor-Critic Methods (e.g., SAC, DDPG):** Will be investigated for their superior sample efficiency, which is critical for online learning in production-like environments.

b). Data Structure Implementation and System Simulation

These tools will be used to build the data structures themselves, the testing environments, and the system-level simulator.

- **Programming Languages:**
    - **Python:** Serves as the main "glue" language for integrating C++ data structures with RL libraries via bindings (e.g., pybind11). It will also be used for building the DataStructureEnv wrapper classes.
- **Simulation & Benchmarking Tools:**
    - **Custom Discrete-Event Simulator:** A Python-based simulator will be developed to model the passage of time and resource contention (CPU cycles, memory allocation) for the system-level experiments in RO2. This allows for controlled and reproducible experiments without the overhead of a full database deployment.

c). Performance Analysis and Evaluation

A rigorous evaluation requires tools for monitoring, profiling, and statistical analysis.

- **Data Analysis and Visualization:**
    - **Pandas & NumPy:** For processing, cleaning, and aggregating the large volumes of performance data generated during experiments.
    - **Matplotlib & Seaborn:** For creating publication-quality graphs to visualize learning curves, latency distributions (CDFs), and performance comparisons.

d). Experimentation and Reproducibility

To ensure robust and trustworthy results, the following practices and tools will be adopted.

- **Experiment Orchestration:**
    - **Weights & Biases (W&B):** This platform will be used to track hyperparameters, code versions, and output metrics from thousands of training runs. It is indispensable for systematically comparing different RL algorithms, reward functions, and coordination strategies.
- **Statistical Techniques:**
    - **Confidence Intervals & Hypothesis Testing:** Performance results will be reported with 95% confidence intervals calculated over multiple independent runs.

Statistical significance testing (e.g., paired t-tests) will be used to validate that performance improvements are not due to random chance.

By leveraging this comprehensive toolkit, the research will be conducted in a systematic, scalable, and scientifically rigorous manner, ensuring that the findings are both valid and impactful.

5. Expected Outcome

This research is expected to yield a novel, publicly available Unified Reinforcement Learning Framework for Data Structures (URL-DS) that demonstrably enables diverse data structures to autonomously self-adapt, achieving superior and more robust performance in dynamic environments compared to static and state-of-the-art adaptive counterparts. Furthermore, it will produce the first formal analysis and a set of proven coordination mechanisms—a meta-controller and a multi-agent RL strategy—to manage the system-level interactions of multiple self-adaptive structures, thereby ensuring global performance optimization and preventing destructive competition. The ultimate outcome is a significant step towards truly autonomous database systems that self-optimize in real-time, reducing administrative overhead and providing predictable performance for modern, data-intensive applications.

6. Conclusion

This synopsis has outlined a comprehensive research plan to address a critical and timely challenge at paradigm of self-adaptive data structures, powered by reinforcement learning, represents a fundamental shift from static design to dynamic, intelligent systems. This research has outlined a comprehensive plan to address the critical limitations of current approaches, which are often narrow in scope, reactive in nature, and blind to system-wide implications. By proposing a unified RL framework (URL-DS), this work aims to democratize the creation of adaptive structures, providing a reusable blueprint for transforming classic data organization models into autonomous, learning entities. Furthermore, by pioneering coordination mechanisms for multi-structure environments, the research confronts the complex, real-world challenge of managing interdependent adaptive components, moving beyond isolated performance gains to achieve robust, global system optimization. The successful execution of this research will not only deliver novel algorithms and a practical software framework but will also establish a foundational

methodology for building the next generation of self-managing, high-performance data-intensive systems capable of thriving in the face of constant change.

This work contends that the foundation of any successful data infrastructure is not laid in code, but in the clarity and completeness of its requirements. This thesis will provide the tools and techniques to build that foundation more effectively for the data-driven challenges of tomorrow.

References

[1]. Al-Mamun, A., Hossain, M. T., Kibriya, M. S., & Khan, Z. R. (2025). AI+R-Tree: A machine learning augmented R-Tree for spatial queries. arXiv preprint, arXiv:2502.09937. https://arxiv.org/abs/2502.09937

[2]. B. Verma *et al.*, "EvolveQS: Queue Structures for Adaptive Concurrency," *Proc. ACM PPoPP*, 2025. doi: 10.1145/3616715.3617651

[3]. D. Wang, N. Gao, D. Liu, J. Li and F. L. Lewis, "Recent Progress in Reinforcement Learning and Adaptive Dynamic Programming for Advanced Control Applications," in *IEEE/CAA Journal of Automatica Sinica*, vol. 11, no. 1, pp. 18-36, January 2024, doi: 10.1109/JAS.2023.123843

[4]. Ding, H., Liu, K., Zhang, Y., & Li, C. (2020). ALEX: An updatable adaptive learned index. Proceedings of the ACM Special Interest Group on Management of Data (SIGMOD), 967–981. https://doi.org/10.1145/3318464.3389711

[5]. Gu, T., Xu, H., & Wang, X. (2024). RLR-Tree: A reinforcement learning-based R-Tree for spatial data. Journal of Big Data, 11(1), Article 120. https://doi.org/10.1186/s40537-024-01025-1

[6]. Heidari, A., Ahmadi, A., & Zhang, W. (2024). UpLIF: An updatable self-tuning learned index framework. *arXiv preprint*, arXiv:2408.04113. https://arxiv.org/abs/2408.04113

[7]. Kim, J., Lee, S., & Park, H. (2024). Predictive tree balancing for edge devices. Proceedings of the IEEE International Conference on Distributed Computing Systems (ICDCS), 1201–1211. https://doi.org/10.1109/ICDCS57903.2024.00111

[8]. Li, X., Xu, H., Zhang, J., & Chang, H. (2022). Deep reinforcement learning for adaptive learning systems. Journal of Educational and Behavioral Statistics, 48(2), 220–243. https://doi.org/10.3102/10769986221129847

[9]. Li, Z., Sun, Y., & Wang, J. (2024). Adaptive indexing with online learning. arXiv preprint, arXiv:2405.09817. https://arxiv.org/abs/2405.09817

[10]. Liu, T., Zhang, K., & Wang, Q. (2023). RL-Stack: A reinforcement learning-based stack with dynamic reordering. arXiv preprint, arXiv:2308.12113. https://arxiv.org/abs/2308.12113

[11]. Rahman, A. (2024). Reinforcement Learning in Dynamic Environments: Challenges and Future Directions. *International Journal of Emerging Research in Engineering and Technology*, *5*(2), 1-11. https://doi.org/10.63282/3050-922X.IJERET-V5I2P101

[12]. S. Joshi and A. Mehta, "PredictAVL: Predictive Balancing in AVL Trees," *arXiv preprint*, Jun. 2024. [Online]. Available: https://arxiv.org/abs/2406.07997

[13]. Singh, D., & Yadav, A. (2023). Reinforcement learning for cache-aware BST structures. arXiv preprint, arXiv:2304.06678. https://arxiv.org/abs/2304.06678

[14]. Shoujiang Ruan, Kebin Lu, Adaptive deep reinforcement learning for personalized learning pathways: A multimodal data-driven approach with real-time feedback optimization, Computers and Education: Artificial Intelligence, Volume 9, 2025, 100463, ISSN 2666-920X, https://doi.org/10.1016/j.caeai.2025.100463.

[15]. Wang, K., Li, Z., & Chen, H. (2025). LITune: A reinforcement learning-based tuning framework for learned index structures. arXiv preprint, arXiv:2502.05001. https://arxiv.org/abs/2502.05001

[16]. Wang, T., Liang, L., Yang, G., Heinis, T., & Yoneki, E. (2025). A new paradigm in tuning learned indexes: A reinforcement learning enhanced approach. *arXiv preprint*, arXiv:2502.05001. https://arxiv.org/abs/2502.05001

[17]. X. Tang, J. Xu and S. Wang, "RoMA: Resilient Multi-Agent Reinforcement Learning with Dynamic Participating Agents," *2023 IEEE 12th International Conference on Cloud Networking (CloudNet)*, Hoboken, NJ, USA, 2023, pp. 247-255, doi: 10.1109/CloudNet59005.2023.10490082.

[18]. Yan, Y., Li, J., & Zaggia, C. (2025). The analysis of deep reinforcement learning for dynamic graphical games under artificial intelligence. Scientific Reports, 15(1), Article 23133. https://doi.org/10.1038/s41598-025-05192-w

[19]. Yuan, M., Li, B., Jin, X., & Zeng, W. (2025). Adaptive data exploitation in deep reinforcement learning. *arXiv preprint*, arXiv:2501.12620. https://arxiv.org/abs/2501.12620

[20]. Zhang, J., Zhang, X., & Li, Y. (2025). Self-adapting language models (SEAL). *Synced Review* (Reporting on MIT Research). https://syncedreview.com/2025/06/16/mit-researchers-unveil-seal-a-new-step-towards-self-improving-ai/

[21]. Zhu, Q., Fan, L., & Weng, N. (2024). Advancements in point cloud data augmentation for deep learning: A survey. *Pattern Recognition*, *153*, 110532. https://doi.org/10.1016/j.patcog.2024.110532